

מבחן SQL מתקדם

סיפור רקע – “מאגר המכירות” (OPERATION SALES COVER) / מבצע כיסוי מכירות

תאריך האירוע הראשוני: ינואר 2026
מיקום ראשוני: קראקס, ונצואלה (אזור התעשייה של פטארה, מבנה ישן שהוסב למרכז נתונים “פרטי”)
גורם מגלה: תחנת CIA בבוגוטה, בשיתוף עם סוכנים מקומיים של ה-DEA שחיפשו רשתות הלבנת הון של קרטלי סמים בשיתוף עם גורמים איראניים.

בסוף 2025, בעקבות עלייה חדה בפעילות מודיעינית איראנית-חיזבאללה באמריקה הלטינית, ה-CIA הפעילה מעקב רחב על שרתים וירטואליים ופיזיים באזור קראקס. אחד השרתים – שנרשם תחת חברת “Ventas Globales C.A.” (חברת “מכירות גלובליות” – שם גנרי כמעט עד כדי חשדנות) – **הכיל מאגר נתונים שנראה תמים לחלוטין: מערכת ניהול מכירות ומשלוחים קלאסית.**

המאגר הועבר ליחידת הסייבר של **המוסד**, ואתם מצוות התחקור הטכני שנבחר לבצע את הניתוח הראשוני כחלק משיתוף הפעולה המודיעיני בין ארה”ב לישראל.

המשימה שלכם:

לתחקר את המאגר באמצעות **שאלות SQL** ו-**נקודות קצה (FastAPI)** שתקבלו בשלד המערכת, ולהפיק ממנו תובנות שיאפשרו להבין:

- מי הגורמים המעורבים
- מה הדפוסים החריגים
- ואיפה “מסתתר” המידע הרגיש בתוך פעילות שנראית רגילה

המטרה שלכם היא אחת בלבד:

לממש שאלות SQL ולהחזיר תוצאות דרך נקודות קצה קיימות.

הקדמה – מבנה הפרויקט ואופן העבודה

במבחן זה אתם מקבלים **שלד מערכת Backend מוכן**, המבוסס על **FastAPI** ומסד נתונים **MySQL**. המערכת נבנתה מראש כך שתדמה סביבת עבודה אמיתית בצוות, שבה אתם נכנסים למערכת קיימת ואתם נדרשים **להשלים לוגיקה חסרה – מבלי לשנות את התשתית.**

מבנה הפרויקט (ברמה רעיונית)

הפרויקט מחולק לשכבות ברורות:

שכבת תשתית – INFRASTRUCTURE האחראית על:

- חיבור ל-MySQL
- יצירת בסיס הנתונים
- טעינת הטבלאות
- הרצת השרת

שכבה זו כבר כתובה ומוכנה, ואינה חלק מהמבחן.

שכבת DAL - DATA ACCESS LAYER

- מכילה פונקציות ריקות שעליכם לממש.
- כאן נכתבות שאילתות ה-SQL בלבד.

שכבת API - FAST API ROUTES

- מכילה נקודות קצה ריקות.
- כאן תחברו בין ה-Endpoint לבין הפונקציה המתאימה מה-DAL.

חוקים חשובים (חובה לקרוא)

חל איסור

- לשנות קבצים שקשורים ל-Docker
- לשנות קבצי חיבור למסד הנתונים
- לשנות קבצי init של בסיס הנתונים
- להשתמש ב-ORM
- להוסיף ספריות חיצוניות
- אין לכתוב SQL מחוץ ל-dal.py

מה עליכם לעשות

- לכתוב SQL רק בתוך הפונקציות ב-`dal.py`
- לממש את הלוגיקה של כל Endpoint ב-`main.py`
- כל Endpoint מחזיר JSON תקני

אתחול אוטומטי של המערכת

בעת הרצת השרת:

- החיבור למסד הנתונים נוצר אוטומטית
- בסיס הנתונים והטבלאות נוצרים אוטומטית
- אין צורך בהרצה ידנית של סקריפטים
- אין צורך בהגדרות נוספות מצדכם


אם השרת עולה – סימן שהכול מוכן לעבודה.

איך מתחילים לעבוד על הפרויקט – שלב אחרי שלב

כדי להתחיל את המבחן, תקבלו מאיתנו קוד מוכן מראש שנמצא ב-GitHub. הקוד כולל את כל התשתית הדרושה (שרת, חיבור למסד נתונים, Docker וכו'), ואתם נדרשים רק להשלים את הלוגיקה החסרה בהתאם לשאלות המבחן.

חשוב מאוד – אין לבצע `GIT CLONE` לריפוזיטורי הזה

במקום זאת, עליכם לבצע **Fork** – כלומר ליצור עותק אישי של הפרויקט תחת חשבון ה-GitHub שלכם.

 קישור לריפוזיטורי:

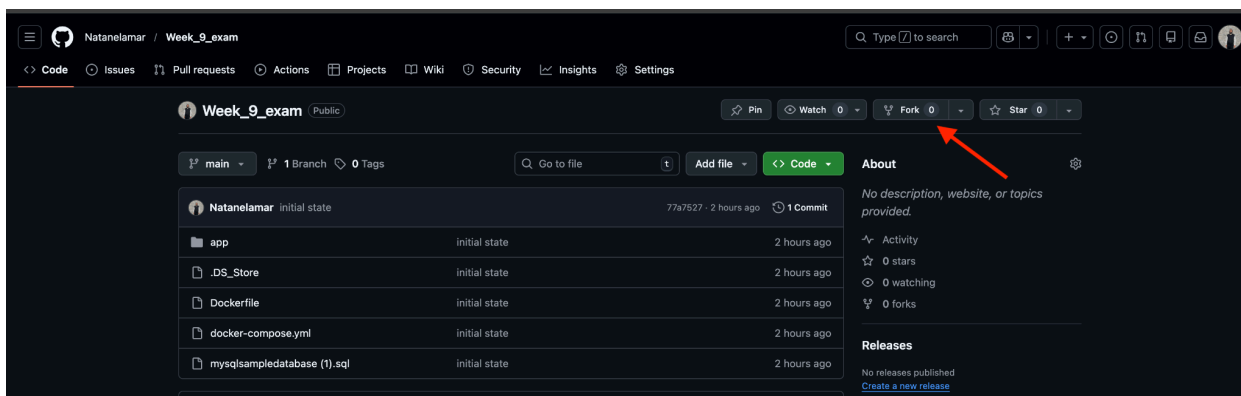
https://github.com/Natanelamar/Week_9_exam

שלב 1 – FORK לריפוזיטורי

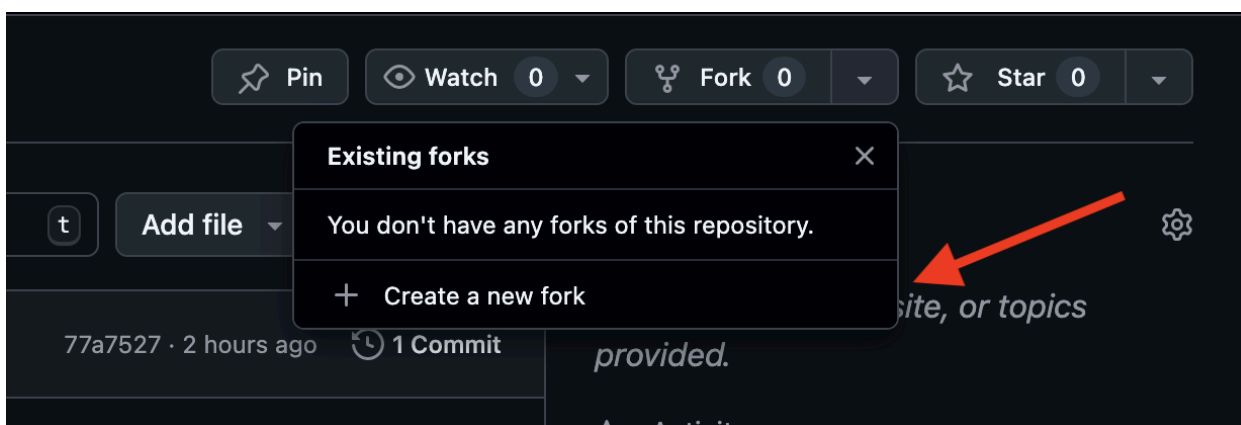
1. היכנסו לקישור של הפרויקט:

https://github.com/Natanelamar/Week_9_exam

2. לחצו על כפתור **Fork** (בצד ימין למעלה)



3. תפתח לשונית ליצירת Fork חדש לחצו עליה



4. בחרו את המשתמש שלכם בתור הבעלים על העותק החדש

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

Choose an owner / Week_9_exam

Search owners ; their upstream repository. You can customize the name to

Natanelamar (repository already exists)

InnovatorsHub1

☒ Copy the `main` branch only

Contribute back to Natanelamar/Week_9_exam by adding your own branch. [Learn more.](#)

Create fork

5. לאחר הפעולה, ייווצר ריפוזיטורי חדש תחת החשבון שלכם בשם: week_15_DBSQL

מעכשיו – זה הפרויקט שלכם.

שלב 2 – עבודה מקומית

לאחר שביצעתם Fork, ניתן לעבוד בצורה רגילה עם הריפוזיטורי שלכם.

1. היכנסו לריפוזיטורי **שנוצר אצלכם**
2. העתיקו את כתובת ה-HTTPS (Clone)
3. במחשב שלכם, פתחו טרמינל והריצו:

```
<git clone <YOUR_FORK_URL>
```

(כאשר במקום <YOUR_FORK_URL> אתם רושמים את הקישור לריפו שלכם בפועל.)

4. היכנסו לתיקיית הפרויקט:
- ```
cd Week_9_exam
```

5. מכאן ואילך אתם עובדים **רק על הריפוזיטורי שלכם.**

## עבודה עם BRANCHES – חובה ( 10 נקודות )

במבחן זה נדרש לעבוד עם Git בצורה נקייה ומקצועית, בהתאם לסטנדרטים מקובלים בתעשייה.

לאחר ביצוע ה-Fork לריפוזיטורי, עליכם להגדיר מבנה Branches ברור ומסודר שכוללים את הבראנצ'ים הבאים:

### MAIN

- מכיל את הקוד הסופי לבדיקה
- רק Branch זה ייבדק

### DEV

- Branch עבודה ראשי
- כל הפיתוח מתבצע דרכו
- בסוף העבודה, יש למזג (merge) אותו ל-main

### BRANCH לפיצ'ר השאילתות

- מימוש שאילתות ה-SQL
- עבודה על dal.py ו-main.py
- מומלץ (אך לא חובה) ליצור Branch נוסף עבור כל שאילתה - זהו הרגל עבודה נכון ומקצועי, המדמה עבודה בצוות אמיתי.

### K8S ייעודי ל-YAML

- Branch נפרד שמיועד אך ורק לעבודה על קבצי ה-YAML ופריסת OpenShift
- מטרתו להפריד בין:
  - פיתוח קוד אפליקציה
  - פיתוח תשתיות ופריסה

## תהליך עבודה מומלץ

- עבודה שוטפת מתבצעת על Branches ייעודיים
- מיזוג (merge) ל-dev
- בסיום כל המבחן:
  - מיזוג dev אל main
  - main חייב להכיל את הפתרון המלא והסופי

## בדיקה וניקוד

- ייבדק שכל ה-Banches קיימים
- ייבדק כי הקוד הסופי נמצא ב-main
- עבודה מסודרת עם Branches היא חלק מהציון הכולל

## שאלות SQL וניתוח נתונים ( 80 נקודות – 8 נקודות פר שאלה )

בשלב זה עליכם לתחקר את מאגר הנתונים שהועבר אליכם, באמצעות שאלות SQL שתממשו במערכת הקיימת.

לכל שאלה:

- קיימת פונקציה ריקה בשכבת ה-DAL
- קיימת נקודת קצה (Endpoint) מתאימה
- עליכם לממש את השאלה ולחבר אותה ל-Endpoint הרלוונטי

### **שאלה 1 – לקוחות עם מסגרת אשראי חריגה**

החזירו את שמות הלקוחות ומסגרת האשראי שלהם עבור לקוחות אשר:

- מסגרת האשראי שלהם נמוכה מ-10,000 או גבוהה מ-100,000

**פלט נדרש:**

- שם הלקוח
- מסגרת אשראי

### **שאלה 2 – הזמנות ללא הערות**

החזירו את מספרי ההזמנות וההערות עבור הזמנות שאין להן הערות כלל (ערך NULL)

**פלט נדרש:**

- מספר הזמנה
- הערות
- מיון: לפי תאריכים של ההזמנות

### **שאלה 3 – חמשת הלקוחות האחרונים**

החזירו את חמשת הלקוחות האחרונים לפי משרד כפי שהם מופיעים בטבלת הלקוחות.

**פלט נדרש:**

- שם הלקוח
- שם משפחה של איש הקשר
- שם פרטי של איש הקשר
- מיון לפי שם משפחה

### **שאלה 4 – סיכום תשלומים כולל, תשלום מינימלי ומקסימלי**

החזירו:

- את סכום התשלומים הכולל



- ואת ממוצע סכום התשלומים

עבור כלל התשלומים במערכת.

**פלט נדרש:**

- סכום תשלומים כולל
- ממוצע תשלום
- תשלום מינימלי
- תשלום מקסימלי

### **שאלה 5 – עובדים ומספרי טלפון של משרדים**

החזירו עבור כל עובד:

- שם פרטי
- שם משפחה
- מספר הטלפון של המשרד שבו הוא עובד

**פלט נדרש:**

- שם פרטי
- שם משפחה
- טלפון משרד

### **שאלה 6 – לקוחות ותאריכי משלוח**

החזירו את שמות הלקוחות ואת תאריכי המשלוח של ההזמנות שלהם וממוצע של כמות ההזמנות הכללית .

יש לכלול גם לקוחות שלא ביצעו הזמנות כלל.

**פלט נדרש:**

- שם הלקוח
- תאריך משלוח

- ממוצע כמות של הזמנה הכללי

### **שאלה 7 – כמות פריטים בכל הזמנה**

החזירו את שמות הלקוחות ואת הכמות שהוזמנה בכל הזמנה.

**פלט נדרש:**

- שם הלקוח
- כמות שהוזמנה
- מיון לפי שם הלקוח

### **שאלה 8 – לקוחות חשודים לפי שם משפחה**

החזירו:

- שם הלקוח
- שם איש המכירות
- סכום התשלומים הכולל

עבור לקוחות ששם הפרטי של איש הקשר שלהם מכיל את המחרוזת 'Mu' או 'ly'.

**פלט נדרש:**

- שם הלקוח
- שם איש המכירות
- סכום תשלומים כולל
- יש למיין את התוצאות לפי סכום התשלומים הכולל בסדר יורד.

## **דגשים כלליים**

- יש להחזיר נתונים מדויקים בהתאם לפלט הנדרש
- אין חשיבות לשמות העמודות, אלא לערכים
- שימוש נכון ב-JOIN הוא חלק מהפתרון

- סדר התוצאות חשוב רק כאשר צוין במפורש

## פריסה לאופן שיפט ( 8 נקודות )

בשלב זה עליכם לפרוס את האפליקציה שבניתם לסביבת OpenShift, באמצעות קבצי YAML שתכתבו בעצמכם.

## קבצי YAML – מה נדרש מכם

עליכם לכתוב את כל קבצי ה-YAML הדרושים על מנת:

- להריץ את אפליקציית ה-FastAPI
- להריץ מסד נתונים MySQL
- לאפשר תקשורת בין האפליקציה למסד הנתונים
- לחשוף את האפליקציה החוצה כך שניתן יהיה לגשת אליה מהדפדפן

מספר קבצי YAML כולל:

ציפייה של לפחות 4 קבצים (בהתאם לפתרון שלכם)

אין פירוט אילו משאבים בדיוק נדרשים.  
עליכם להבין לבד אילו רכיבים יש להגדיר כדי שהמערכת תעבוד מקצה לקצה.

## צילומי מסך ( 2 נקודות )

בסיום הפריסה, עליכם להעלות 2 צילומי מסך:

### 1. כל ה-Pods רצים

a. צילום מסך המראה שכל ה-Pods בפרויקט במצב Running.

b. יש לוודא שניתן לראות את שמות ה-Pods והסטטוס שלהם.

## **2. האפליקציה רצה בדפדפן**

צילום מסך של האפליקציה פתוחה בדפדפן.

a. יש להציג את שורת ה-URL המלאה

b. יש לראות בבירור את הכתובת שנוצרה על-ידי OpenShift

c. יש לגשת ל-Route של האפליקציה (לא localhost)